МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

ОТЧЕТ

по лабораторной работе №1

по дисциплине

Шаблоны проектирования ПО

РУКОВОДИТЕЛЬ:

_____ ___Жевнерчук Д. В.___

(подпись) (фамилия, и.,о.)

СТУДЕНТЫ: ____Дёмин Д. И.____

__Карпычева А. Ю.__

_____ ____Пигасин Д. А.___

(подпись) (фамилия, и.,о.)

_____18-ИВТ-1_____

(шифр группы)

Работа защищена «___» _____

С оценкой _____

Нижний Новгород 2019

Вариант 6

Реализуйте систему генерации шаблона html страницы по описанию:

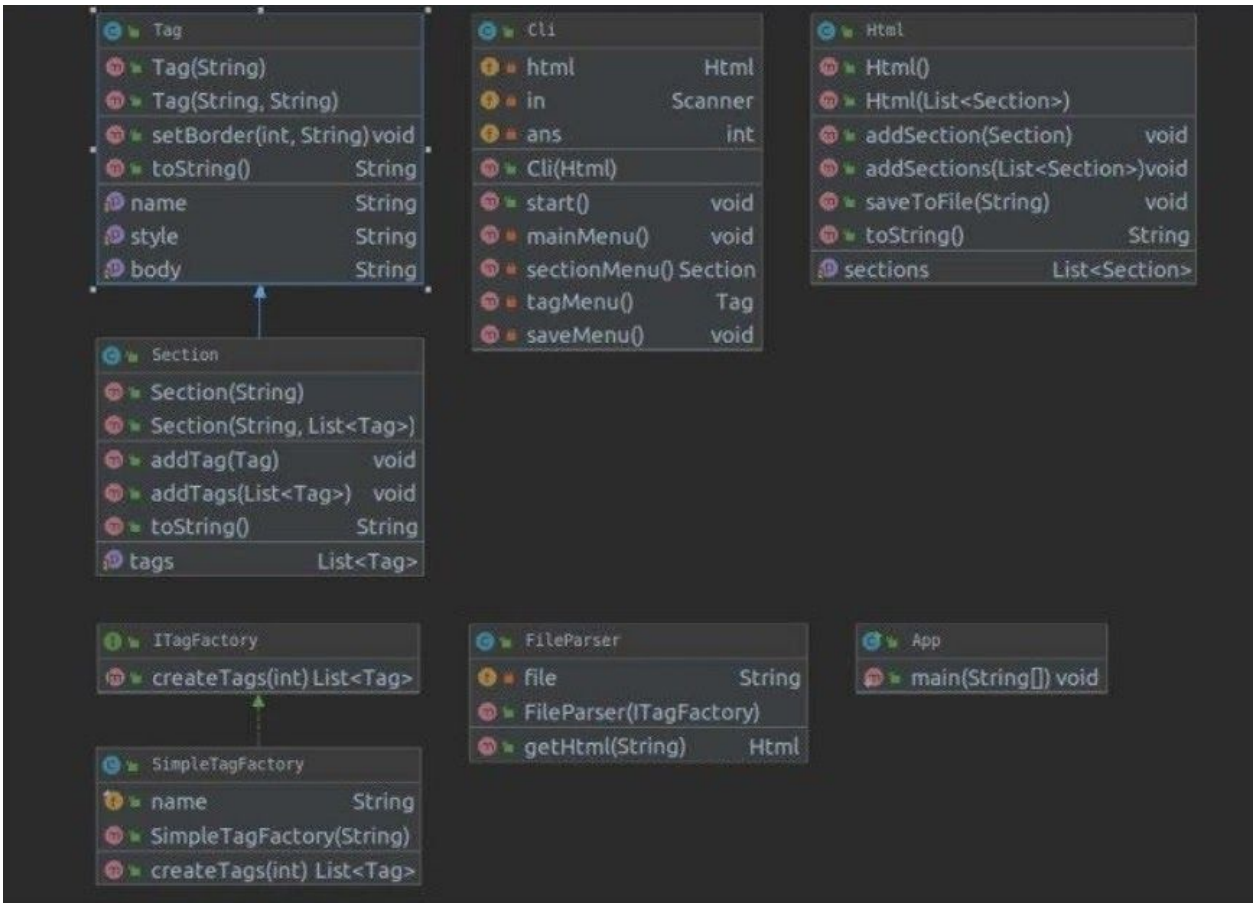
1. Состава разделов (header, section1, ... , section_n, footer)
2. Структуры каждой секции (количество блоков div)

Добавьте опциональную возможность выделять определенный раздел, блок рамкой

Сгенерированный html код вводите в консоль.

Реализация

UML схема



При написании программы мы руководствовались основными принципами ООП, а также использовали паттерны проектирования GOF там, где это уместно.

Например, конструктор класса FileParser принимает экземпляр фабрики, реализующей интерфейс ITagFactory.

Такой подход позволит нам изменять логику создания тэгов FileParser’ом, не переписывая то, что уже было сделано.

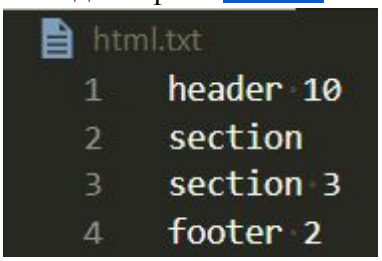
Для этого нужно описать новый класс, реализующий интерфейс ITagFactory и содержащий нужный нам алгоритм.

Как использовать

1. Описываем структуру страницы во внешнем файле.
    - Сначала указывается имя секции, количество вложенных блоков.
    - Одна секция - одна строка
2. Запускаем программу
3. Выделяем нужные блоки рамкой (опционально)
4. Сохраняем полученный шаблон (опционально)

Пример

1. Создаем файл html.txt:



2.Запускаем программу.

Код программы

app.App.java

```
package app;

import html.FileParser;
import html.SimpleTagFactory;
import html.base.Html;

public class App {
	public static void main(String[] args) throws Exception {
	FileParser txt = new FileParser(new SimpleTagFactory("div"));
	Html html = txt.getHtml("html.txt");

	Cli app = new Cli(html);
	app.start();
	}

}
```

app.Cli.java

```
package app;

import java.io.IOException;
import java.util.Scanner;

import html.base.Html;
import html.base.Section;
import html.base.Tag;

public class Cli {
```

```
private Html html;
private Scanner in = new Scanner(System.in);
private int ans;

public Cli(Html html) {
this.html = html;
}

public void start() throws IOException {
do {
mainMenu();
} while (ans != 3);

saveMenu();
in.close();
}

private void mainMenu() {
System.out.println(html);
System.out.println("1.Add section border\n2.Add tag border\n3.Exit");
System.out.print(">>>");
ans = in.nextInt();

if (ans == 1) {
Section section = sectionMenu();
section.setBorder(5, "black");
}

if (ans == 2) {
Tag tag = tagMenu();
tag.setBorder(5, "black");
```

```
}
}

private Section sectionMenu() {
Section[] sections = html.getSections();

for (int i = 0; i < sections.length; i++) {
System.out.println(i + "." + sections[i].getName());
}

System.out.print(">>>");
return sections[in.nextInt()];
}

private Tag tagMenu() {
Section section = sectionMenu();
Tag[] tags = section.getTags();

for (int i = 0; i < tags.length; i++) {
System.out.println(i + "." + tags[i]);
}

System.out.print(">>>");
return tags[in.nextInt()];
}

private void saveMenu() throws IOException {
System.out.println("\nSave the file as index.html?\n1.Yes\n2.No");
ans = in.nextInt();

if (ans == 1)
```

```
        html.saveToFile("index.html");
        }

}
```

html.base.Tag.java

```
package html.base;

public class Tag {
        protected String name;
        protected String body;
        protected String style;

        public Tag(String name) {
        this.name = name;
        }

        public Tag(String name, String body) {
        this.name = name;
        this.body = body;
        }

        public String getName() {
        return name;
        }

        public void setName(String name) {
        this.name = name;
        }

        public String getBody() {
```

```
return body;
}

public void setBody(String body) {
this.body = body;
}

public String getStyle() {
return style;
}

public void setStyle(String style) {
this.style = style;
}

public void setBorder(int thickness, String color) {
setStyle("border:" + thickness + "px solid " + color);
}

@Override
public String toString() {
String tagHead = "<" + name;
String tagBody = "";
String tagTail = "</" + name + ">";

if (body != null)
tagBody += body;

if (style != null)
tagHead += " style='" + style + "'";
```

```
        tagHead += ">";

        return tagHead + tagBody + tagTail;
        }

}
```

html.base.Section.java

```
package html.base;

import java.util.LinkedList;
import java.util.List;

public class Section extends Tag {
        private List<Tag> tags = new LinkedList<Tag>();

        public Section(String name) {
        super(name);
        }

        public Section(String name, List<Tag> tags) {
        super(name);
        this.tags = tags;
        }

        public Tag[] getTags() {
        return tags.toArray(new Tag[tags.size()]);
        }

        public void setTags(List<Tag> tags) {
        this.tags = tags;
        }
```

```
    public void addTag(Tag tag) {
    tags.add(tag);
    }

    public void addTags(List<Tag> tags) {
    for (Tag tag : tags) {
    addTag(tag);
    }
    }

    @Override
    public String toString() {
    String tagHead = "<" + name;
    String tagBody = "";
    String tagTail = "\n</" + name + ">";

    if (style != null)
    tagHead += " style='" + style + "'";

    tagHead += ">";

    for (Tag tag : tags) {
    tagBody += "\n\t" + tag;
    }

    return tagHead + tagBody + tagTail;
    }

}
```

html.base.Html.java

```
package html.base;

import java.io.FileWriter;
import java.io.IOException;
import java.util.LinkedList;
import java.util.List;

public class Html {
	private List<Section> sections = new LinkedList<Section>();

	public Html() {
	}

	public Html(List<Section> sections) {
	this.sections = sections;
	}

	public Section[] getSections() {
	return sections.toArray(new Section[sections.size()]);
	}

	public void setSections(List<Section> sections) {
	this.sections = sections;
	}

	public void addSection(Section section) {
	sections.add(section);
	}

	public void addSections(List<Section> sections) {
```

```
        for (Section section : sections) {
        addSection(section);
        }
        }

        public void saveToFile(String fileName) throws IOException {
        FileWriter file = new FileWriter(fileName);
        file.write(toString());
        file.close();
        }

        @Override
        public String toString() {
        String html = "";
        for (Section section : sections) {
        html += section + "\n\n";
        }
        return html;
        }

}
```

html.FileParser.java

```
package html;

import java.io.FileNotFoundException;
import java.io.FileReader;
import java.util.LinkedList;
import java.util.List;
import java.util.Scanner;

import html.base.Html;
```

```
import html.base.Section;

public class FileParser {
	private ITagFactory tagFactory;

	public FileParser(ITagFactory tagFactory) {
	this.tagFactory = tagFactory;
	}

	public Html getHtml(String file) throws FileNotFoundException {
	Scanner scan = new Scanner(new FileReader(file));
	List<Section> sections = new LinkedList<Section>();

	while (scan.hasNextLine() & scan.hasNext()) {
	Scanner line = new Scanner(scan.nextLine());
	Section section = new Section(line.next());

	if (line.hasNextInt())
		section.addTags(tagFactory.createTags(line.nextInt()));

	sections.add(section);
	}
	scan.close();

	return new Html(sections);
	}
}
```

html.ITagFactory.java

```
package html;

import java.util.List;
```

```
import html.base.Tag;

public interface ITagFactory {
	public List<Tag> createTags(int count);
}
```

html.SimpleTagFactory.java

```
package html;

import java.util.LinkedList;
import java.util.List;

import html.base.Tag;

public class SimpleTagFactory implements ITagFactory {
	final public String name;

	public SimpleTagFactory(String name) {
	this.name = name;
	}

	public List<Tag> createTags(int count) {
	List<Tag> tags = new LinkedList<Tag>();
	for (int i = 0; i < count; i++) {
	tags.add(new Tag(name, name + "_" + i));
	}
	return tags;
	}

}
```